

Nástroj pro tvorbu rozvrhů Rozvrhář

Dokumentace maturitní práce z programování

Šimon Vyhnis

# Zadání maturitní práce

Cílem práce je vytvořit v programovacím jazyce C# s použitím frameworku ASP.NET webovou aplikaci pro tvorbu školních rozvrhů. Aplikace uživateli na základě předem nahraných dat napovídá, které hodiny může do rozvrhu přidávat, aniž by s něčím kolidovaly. Do aplikace půjdou nahrávat data vyexportovaná ze systému Bakaláři. Data budou uložena v SQL databázi.

Teoretická část: Relační databáze

Prohlašuji, že jsem na práci pracoval samostatně pouze za pomoci použitých zdrojů a že v práci i v dokumentaci jasně vymezuji, které části kódů jsou mým originálním dílem, které jsou upravenou verzí a které jsou převzaty v plném rozsahu.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

podpis žáka

Obsah

[Zadání maturitní práce 2](#_Toc194006886)

[1. Teoretická část – Relační databáze 6](#_Toc194006887)

[1.1 Princip integrity 6](#_Toc194006888)

[1.2 Klíče a jejich typy 6](#_Toc194006889)

[1.3 Vazby mezi tabulkami 7](#_Toc194006890)

[1.4 Structured Query Language (SQL) 8](#_Toc194006891)

[1.4.1 Data definition language (DDL) 8](#_Toc194006892)

[1.4.2 Data manipulation language (DML) 8](#_Toc194006893)

[1.4.3 Data control language (DCL) 8](#_Toc194006894)

[1.5 Objektově relační mapování 8](#_Toc194006895)

[1.6 Výhody a nevýhody relační databáze 8](#_Toc194006896)

[2. Cíle práce 9](#_Toc194006897)

[3. Způsoby řešení a použité postupy 10](#_Toc194006898)

[3.1 Databázové schéma 10](#_Toc194006899)

[3.2 Parsování CSV 11](#_Toc194006900)

[3.3 Modals 11](#_Toc194006901)

[3.4 Přihlašování 11](#_Toc194006902)

[3.5 TableData 11](#_Toc194006903)

[3.5.1 GetTimetable 12](#_Toc194006904)

[3.5.2 GetAvailableLessons 12](#_Toc194006905)

[4. Programovací prostředky 13](#_Toc194006906)

[4.1 Programovací jazyk a framework 13](#_Toc194006907)

[4.2 Databáze a objektově relační mapování 13](#_Toc194006908)

[5. Zhodnocení dosažených výsledků 13](#_Toc194006909)

[5.1 Míra splnění zadání 13](#_Toc194006910)

[5.2 Možnosti rozšíření 13](#_Toc194006911)

[6. Instalace 14](#_Toc194006912)

[6.1 Návod pro instalaci 14](#_Toc194006913)

[6.2 Systémové nároky 14](#_Toc194006914)

[7. Ovládání 15](#_Toc194006915)

[8. Seznam použitých zdrojů 16](#_Toc194006916)

[8.1 Seznam informačních zdrojů 16](#_Toc194006917)

[8.2 Seznam zdrojů kódu 17](#_Toc194006918)

# 1. Teoretická část – Relační databáze

Relační databáze jsou v současné době nejrozšířenějším způsobem ukládání dat ve větších informačních systémech. Princip relační databáze formuloval na počátku 70. let informatik Edgar Frank Codd na základě matematické teorie relačních množin. S prvními implementacemi přišly v 80. letech firmy Oracle a IBM a relační databáze se záhy ukázaly jako lepší řešení než dříve používané síťové a hierarchické databáze.

Základním prvkem relační databáze je relace – dvojrozměrná tabulka, která obsahuje jednotlivé záznamy (řádky). Každý sloupec v tabulce, atribut, má definovaný název a datový typ, data jednotlivých záznamů odpovídají struktuře definované atributy.

|  |  |  |  |
| --- | --- | --- | --- |
| id [int] | title [varchar 50] | name [varchar 200] | hours [int] |
| 1 | Mgr. | Monika Boušková | 20 |
| 2 | Ing. | Karel Kratochvíl | 15 |
| 3 | RnDr. | Stanislav Gottwald | 20 |

*Jednoduchý přiklad databázové tabulky učitelů. Tabulka má 4 atributy (id, titul, jméno a počet hodin) a obsahuje 3 záznamy (Moniku Bouškovou, Karla Kratochvíla a Stanislava Gottwalda).*

## 1.1 Princip integrity

Integrita databáze znamená, že data v ní odpovídají všem definovaným pravidlům a strukturám. Kontrola integrity při manipulaci s daty může probíhat jak přímo v databázi, tak na straně aplikace. Často se oba přístupy kombinují.

## 1.2 Klíče a jejich typy

Atributy nebo skupiny atributů mohou plnit funkci databázových klíčů. Základní typ je **kandidátní klíč**. Ten určuje, že daný atribut (nebo skupina) je pro každý záznam v jedné tabulce unikátní. Každá tabulka může obsahovat jeden nebo více kandidátních klíčů. Kandidátní klíče by měly obsahovat nejnižší možný počet atributů, který zaručí unikátnost.

Jeden z kandidátních klíčů může mít roli **primárního klíče**. Primárním klíčem může být pouze jeden atribut (nikoliv skupina) a slouží k identifikaci záznamu napříč tabulkami. Primární klíče se většinou generují jako speciální atribut přímo v databázi, a to buď sekvenčně (pořadím vytvořeného prvku), nebo pseudonáhodně. Vygenerovaný atribut se většinou nazývá ID.

Ostatní kandidátní klíče mimo primárního se označují jako **alternativní**. Pro propojování tabulek v databázi mezi sebou se používají **cizí klíče**.

## 1.3 Vazby mezi tabulkami

Pro vytvoření složitějších datových struktur lze v relační databázi přidávat vztahy mezi jednotlivými tabulkami. Vztah se tvoří pomocí primárního klíče jedné tabulky uloženého v jiné tabulce, tedy cizího klíče. Existují tři základní typy vztahů: 1:1, 1:n a n:n.

Ve vztahu **1:1** odpovídá každému záznamu z jedné tabulky právě jeden záznam v druhé tabulce. Obě tabulky obsahují cizí klíč.

|  |  |  |
| --- | --- | --- |
| id | title | teacher\_id |
| 1 | 1.A | 1 |
| 2 | 2.A | 3 |
| 3 | 3.A | 2 |

|  |  |  |
| --- | --- | --- |
| id | name | class\_id |
| 1 | Monika Boušková | 1 |
| 2 | Karel Kratochvíl | 3 |
| 3 | Stanislav Gottwald | 2 |

*Příkladem vztahu 1:1 může být např. vazba mezi třídou a třídním učitelem. Obě tabulky mají cizí klíč – teacher\_id a class\_id.*

Ve vztahu **1:n** odpovídá jednomu záznamu z jedné tabulky více záznamů z tabulky druhé. Cizí klíč obsahuje pouze druhá tabulka.

|  |  |  |
| --- | --- | --- |
| id | name | class\_id |
| 1 | Martin Novák | 1 |
| 2 | Petr Stuchlý | 1 |
| 3 | Martina Šikovná | 3 |

|  |  |
| --- | --- |
| id | title |
| 1 | 1.A |
| 2 | 2.A |
| 3 | 3.A |

*Příkladem vztahu 1:n může být vazba mezi třídou a jejími studeny. Cizí klíč (class\_id) mají pouze studenti.*

Nejsložitějším typem vztahu je **n:n**, kde každému záznamu z jedné tabulky odpovídá více záznamů z druhé a zároveň každému záznamu z druhé tabulky může odpovídat více záznamů z první. Pro realizaci takového vztahu se používá propojovací tabulka, ve které se ukládají pouze vztahy záznamů mezi dvěma původními tabulkami.

|  |  |
| --- | --- |
| id | name |
| 1 | Martin Novák |
| 2 | Petr Stuchlý |
| 3 | Ema Šikovná |

|  |  |
| --- | --- |
| group\_id | student\_id |
| 1 | 1 |
| 3 | 1 |
| 3 | 3 |

|  |  |
| --- | --- |
| id | title |
| 1 | Angličtina 1 |
| 2 | Angličtina 2 |
| 3 | Němčina |

*Příkladem vztahu n:n může být vazba mezi studenty a skupinami. Vazby probíhají pomocí tabulky student\_to \_group. Martin Novák je ve skupině angličtina1 a němčina. Zároveň ve skupině němčina je krom Martina také Ema Šikovná.*

## 1.4 Structured Query Language (SQL)

Valná většina relačních databází používá pro vnější přístup standardizovaný jazyk Structured Query Language, který umožňuje načítání a úpravu dat pomocí příkazů (query). Syntaxe příkazů vychází z angličtiny, ale jejich formát je pevně daný. Dělí se na 3 základní skupiny.

### 1.4.1 Data definition language (DDL)

DDL slouží k vytváření a úpravě struktury pro samotná data. Patří sem především příkazy pro vytváření, úpravu a mazání tabulek (CREATE TABLE, ALTER TABLE, DROP TABLE) nebo práci s klíči.

### 1.4.2 Data manipulation language (DML)

DML zastřešuje nejpoužívanější příkazy, protože zprostředkovává manipulaci se samotnými daty. Jak jejich načítání (příkaz SELECT), tak úpravu (příkazy INSERT, UPDATE a DELETE).

### 1.4.3 Data control language (DCL)

DCL hraje důležitou roli především ve větších systémech, umožňuje vytvářet uživatele s různými právy a spouštět databázové transakce.

## 1.5 Objektově relační mapování

Pro přístup do relační databáze se v objektově orientovaném programovaní často využívá systému objektově relačního mapování. Tato technika umožňuje automatickou konverzi mezi databázovými záznamy a instancemi tříd a zajišťuje persistentní uchování dat v systému. Do velké míry také dokáže odstínit programátora od přímé práce s databází, protože se databázové příkazy automaticky generují (na pozadí, nebo už při kompilaci).

## 1.6 Výhody a nevýhody relační databáze

Relační databáze má celou řadu výhod, asi nejdůležitější z nich je vysoká míra organizovanosti a struktury v datech. Díky velkému rozšíření existuje mnoho různých implementací a je velká pravděpodobnost nalezení ideální implementace pro konkrétní aplikaci. Moderní relační databáze také poskytují zabezpečení a stabilitu potřebné pro ukládání citlivých dat.

Mezi nevýhody patří složitější návrh databázového schématu, náročnost na výkon při práci s velkými objemy dat a omezená flexibilita při změnách struktury. Také škálování relačních databází může být složitější ve srovnání s některými nerelačními řešeními.

# 2. Cíle práce

Vytváření rozvrhu na střední škole je poměrně složitý proces, do kterého vstupuje celá řada požadavků a proměnných. Přestože dnes většina škol má data o studentech a učitelích v informačních systémech, neposkytují tyto systémy efektivní nástroj pro vytváření rozvrhu. Cílem této maturitní práce je vytvořit samostatně běžící nástroj, který uživateli poskytne efektivní způsob vytváření rozvrhu.

Tento problém nabízí rámcově dvě možná řešení. První možností je generovat celý rozvrh na základě zadaných dat kompletně automaticky a následně nechat uživatele pouze upravovat vygenerovaný návrh. Druhou možností je uživateli pouze napovídat, které hodiny je možné do rozvrhu přidat, ale nechat samotné přidávání hodin na něm. Já jsem zvolil druhý přístup, protože není možné v rámci programu zpracovat všechny možné specifické požadavky pro rozvrh (zejména při zachování jednoduchého ovládání), a především není možné optimálně vyhodnocovat protichůdné požadavky a nacházet vhodné kompromisy. Program by měl tedy hlídat především kolize, specifické požadavky by měl zpracovávat uživatel.

Předem zřejmá limitace vytváření samostatného nástroje je v nutnosti zadat do programu velké množství dat před samotným začátkem tvorby rozvrhu. Tento problém se pokusím řešit možností importu dat z existujících systémů, ale jedná se spíše o zmírnění problému než o jeho řešení.

Pro využitelnost aplikace jsou podstatné její snadná dostupnost, uživatelská jednoduchost a spolehlivost. Snadnou dostupnost by měl zajistit vývoj aplikace jako webové, důležitá je optimalizace uživatelského rozhraní pro různé velikosti obrazovek. Jednoduchosti ovládání bych rád dosáhl využitím interaktivního vykreslování založeného na komponentech. Ke spolehlivosti by mělo přispět použití velkých a ověřených technologií a knihoven.

# 3. Způsoby řešení a použité postupy

## 3.1 Databázové schéma

A screenshot of a computer

AI-generated content may be incorrect.Základem celého programu jsou data, resp. jejich struktura v databázi. Na obrázku níže je můj návrh schématu před samotným vývojem programu. Výsledná databáze tomuto schématu téměř odpovídá, kromě tabulek out\_day a subject\_to\_room, protože měly sloužit pro funkcionality, které jsem nakonec neimplementoval.

Důležitým hlediskem, které není ve schématu znázorněno je možnost atributů nabývat hodnoty null. To je důležité především u cizích klíčů, respektive při mazání záznamů. Pokud totiž cizí klíč nemůže nabýt hodnoty null, dochází k tzv. kaskádovému mazání, tedy že při smazání záznamu, na který cizí klíč odkazuje, dojde rovněž ke smazání záznamu nesoucího cizí klíč. Toho využívám u předmětů a hodin. Nedává smysl, aby existoval předmět, který nemá učitele nebo skupinu studentů, protože by k němu v programu nebylo možné přistoupit, a proto je žádoucí ho při smazání skupiny, třídy, nebo učitele smazat také. Stejně tak u hodiny nedává smysl, aby existovala bez mateřského předmětu, tudíž je žádoucí ji kaskádovitě smazat. Naopak ke studentovi má uživatel přístup i mimo obrazovku třídy, a proto může zůstat bez třídy (atribut třídy se nastaví na hodnotu null a student zůstane zachován). Specifická je n:n vazba studentů a skupin. V té sice při smazání studenta dochází ke kaskádovému smazání všech záznamů o jeho vztazích, ale samotné skupiny nejsou ovlivněny, stejně tak opačně při mazání skupin.

## 3.2 Parsování CSV

Při přidávání studentů importem (ImportStudentsModal) program nejprve pomocí třídy StreamReader přečte obsah souboru a poté ho zpracovává pomocí standartních metod pro práci s řetězci. Pro každého nalezeného studenta se snaží nalézt třídu odpovídající názvem třídě ze souboru. Před uložením dat se uživateli vypíše informace o tom, kolik uživatelů bylo nalezeno a pro kolik z nich se podařilo najít třídu, aby se minimalizovalo riziko přidání chybně načtených dat do databáze.

## 3.3 Modals

Pro větší přehlednost uživatelského rozhraní neobsahuje aplikace oddělená okna pro přidávání a mazání dat, ale řeší tuto problematiku pouze menšími vyskakovacími okny (modals). Tyto okna jsou zároveň řešena jako generické komponenty AddModal a DeleteModal, do kterých se pouze pomocí parametru vkládá konkrétní formulář pro daný objekt. Díky tomu není nutné zbytečně opakovaně psát kód pro obsluhu vyskakovacího okna.

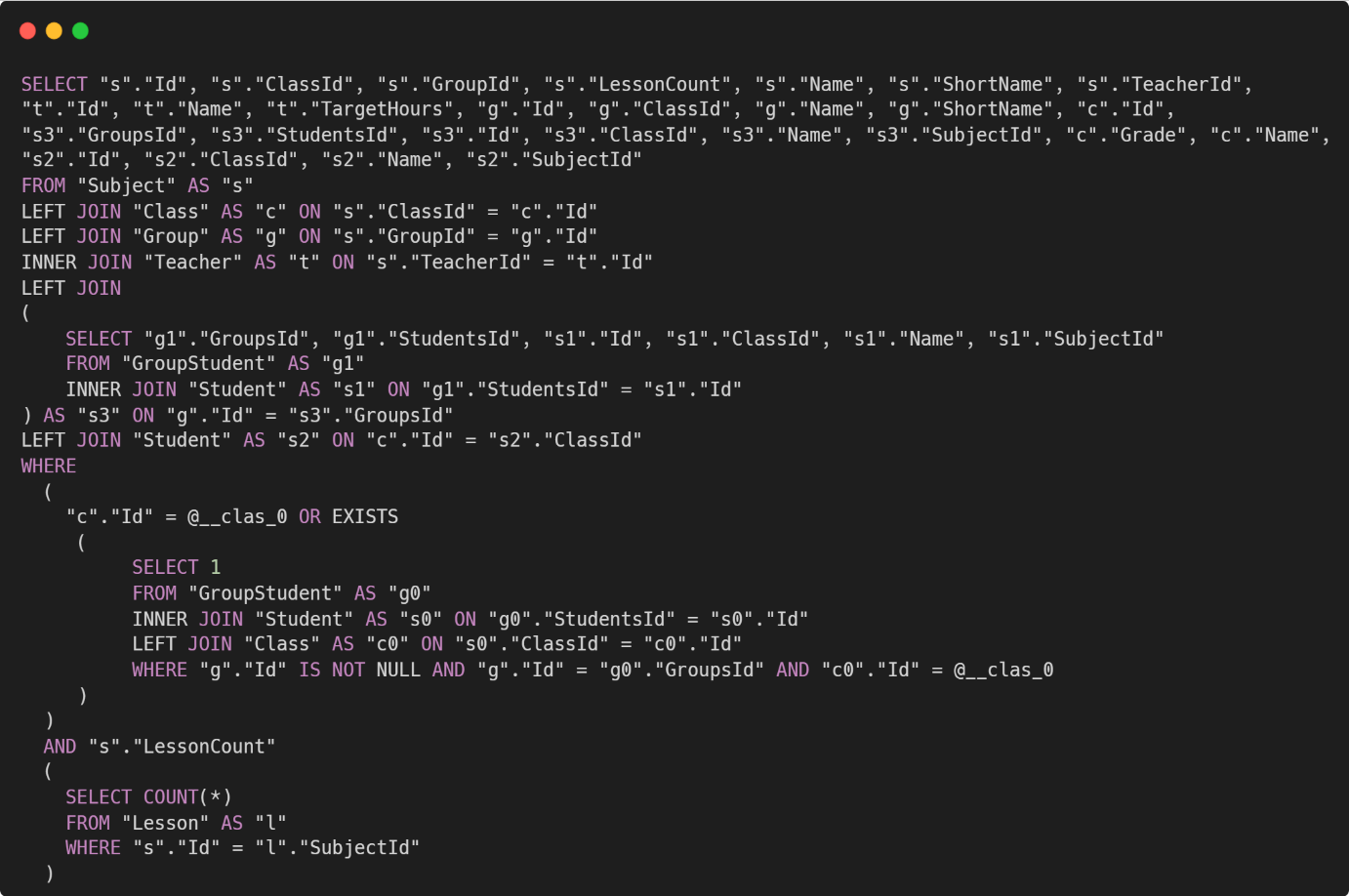
## 3.4 Přihlašování

Pro přihlašování uživatelů využívám funkcionality obsažené v ASP.NET core, které ověřují uživatele pomocí tokenu uloženého v souborech cookie. Zprostředkovávají také hashování a ukládání hesel. Stránky pro přihlašování a odhlašování jsou na rozdíl od ostatních obrazovek aplikace vytvořené jako oddělené stránky, ne jako komponenty, protože pro změnu cookies je třeba poslat nový request (komponenty komunikují pouze pomocí websocketu).

## 3.5 TableData

Algoritmicky asi nejzajímavější částí programu je třída TableData, která zprostředkovává data pro tabulku s rozvrhem a její úpravy. Kvůli většímu rozsahu jsem se rozhodl logiku v TableData oddělit do zvláštního servisu. Využívám pro něj dependency injection (DI), kdy TableData jednoduše přidám mezi servisy při spouštění programu (Program.cs) a následně jen získávám instanci „injektováním“ tam, kde data potřebuji.

### 3.5.1 GetTimetable

Metoda GetTimetable je pomyslným rozcestníkem do metod pro získávání rozvrhu třídy, učitele nebo učebny. V každé z těchto metod se iteruje ve dvou vnořených cyklech přes všechna pole v tabulce rozvrhu. Pro každé pole se načítají všechny odpovídající hodiny. Načítání odpovídajících hodin je zajímavé především u rozvrhu třídy, protože kromě předmětů přímo spadajících pod třídu je nutné zahrnout také všechny předměty skupin, ve kterých je alespoň jeden student z dané třídy.

*Databázový příkaz automaticky generovaný pomocí EF core. Vyhledávání hodin dané třídy tímto způsobem probíhá na více místech v TableData.*

### 3.5.2 GetAvailableLessons

GetAvailableLessons je klíčová metoda, která vrací pouze ty hodiny, které s ničím nekolidují a je tedy možné je přidat do rozvrhu. Podobně jako u GetTimetable se logika mírně liší podle toho, jestli přidáváme hodiny třídě, učiteli, nebo do učebny, ale základní princip je stejný. Vždy se nejprve načtou z databáze všechny předměty, které ještě nemají naplněný cílový počet hodin. Následně se kontroluje, jestli všechny studenti ze třídy (nebo skupiny) předmětu nemají jinou hodinu, jestli je volná (nějaká nebo určitá) učebna a zda má volno učitel daného předmětu. Pro předměty, které všemi kontrolami projdou, se vytvoří objekty nových hodin, připravené pro případné přidání do databáze.

# 4. Programovací prostředky

## 4.1 Programovací jazyk a framework

Program je napsán v programovacím jazyce C# a běží na frameworku ASP.NET core. C# poskytuje moderní přístup k asynchronnímu programování a řešení hodnot null, obě problematiky se v kódu hojně vyskytují. Framework ASP.NET, resp. Blazor, umožňuje vytváření tzv. razor komponent, ve kterých se kombinují html elementy s business logic v rámci jednoho souboru, což výrazně usnadňuje vývoj. Při vývoji jsem využíval IDE Visual Studio 2022, které je přímo vyvinuto pro všechny použité technologie.

## 4.2 Databáze a objektově relační mapování

Program přistupuje k datům výhradně skrz Entity First framework, který umožňuje pokročilé objektově relační mapování, včetně generování velmi složitých databázových příkazů. Jako databázi jsem původně používal Microsoft SQL Server, ale později jsem pro jednodušší lokální instalaci programu zvolil Sqlite.

# 5. Zhodnocení dosažených výsledků

## 5.1 Míra splnění zadání

Požadavky ze zadání byly téměř kompletně splněny. Aplikace využívá zadané programovací prostředky a napovídá uživateli, které hodiny může přidat. Navíc ukazuje, které hodiny ještě zbývají k přidání a umožňuje sledování úvazků pedagogů. Studenty a učitele je možné nahrávat z CSV souborů, jež lze vyexportovat ze systému Bakaláři.

## 5.2 Možnosti rozšíření

Aplikace nabízí celou řadu možných rozšíření, protože při tvorbě rozvrhů je mnoho specifických požadavků, které nejsou implementovány. Se dvěma takovými funkcemi počítá původní databázové schéma. První z nich je možnost zadat do programu dny, ve kterých je nějaký učitel nepřítomný a nemohly by mu tedy být přidávány hodiny na tento den. Druhou je možnost umístění některých předmětů pouze do určitých učeben (např. tělesná výchova pouze v tělocvičně, laboratoře pouze ve specializovaných učebnách apod.).

Aplikace také nekontroluje některé méně pravděpodobné způsoby vzniku kolizí v rozvrhu, především se jedná o zpětnou úpravou dat po začátku vytváření rozvrhu. Například pokud změním učitele nějakého předmětu, není zajištěno, že žádná z jeho dosavadních hodin nebude kolidovat s hodinami, které „získá“ s novým předmětem.

V aplikaci chybí možnosti pro export rozvrhu po jeho vytvoření. Mohl by být exportován jak pro zobrazení (např. pdf), tak pro nahrání do jiného programu. Systém bakaláři ale bohužel umožňuje import varianty rozvrhu pouze v interním formátu s lokálními klíči, tudíž stejně není možné do něj hotový rozvrh importovat.

Uživatelské rozhraní také obsahuje prostor pro zlepšení, např. atraktivnější indikace stavu načítání, přirozenější animace, nástroje vyhledávání nebo nápověda pro nezkušené uživatele. Celý program by také mohl být testován pomocí unit testů, což by pomohlo zlepšit jeho spolehlivost, ale vyžadovalo by to rozsáhlé použití mockování dat.

# 6. Instalace

## 6.1 Návod pro instalaci

Pro lokální spuštění je třeba extrahovat složku TimetableHelper.zip a poté spustit soubor TimetableHelper.exe, který se v ní nachází. Pro správné fungování programu je nutné, aby .exe soubor zůstal ve složce se všemi ostatními extrahovanými soubory, především se složkou wwwroot.

A screen shot of a computer program

AI-generated content may be incorrect.Po spuštění souboru by se mělo otevřít konzolové okno, do nějž program vypíše, na jakém portu běží. Do prohlížeče tedy stačí zkopírovat vypsanou adresu.

## 6.2 Systémové nároky

Hostování aplikace je možné na zařízeních s operačním systémem Windows 7, 8, 10 a 11 a Windows Server, podmínkou je rozhraní .NET Framework verze 4.8 nebo novější. Za použití nástroje Nginx je možné hostování i na některých distribucích linux. Minimální hardware nároky jsou kapacita RAM 1 GB, frekvence procesoru 1kHz a dostupné 4 GB na disku.

Pro klientský přístup k aplikaci jsou podporované prohlížeče Google Chrome, Mozilla Firefox, Microsoft Edge a Apple Safari v nejnovějších verzích. Aplikace je optimalizovaná pro obrazovky o velikosti alespoň 1280x720.

# 7. Ovládání

Aplikace se ovládá jako běžná webová stránka. Mezi jednotlivými obrazovkami lze přepínat pomocí navigačního panelu v levé části. Pro odhlášení a správu uživatelů slouží dvě ikony ve spodní části panelu. Na stránce s rozvrhy lze klikat na všechna pole tabulky rozvrhu a po kliknutí se zobrazí možnosti přidávání a odebírání hodin.

Při vytváření rozvrhu je vhodné držet se následujícího postupu (zjednodušeně popsaného na domovské obrazovce). Na začátku je ideální vytvořit třídy, do kterých se poté studenti automaticky přiřadí při importu. Při nahrávání CSV souboru se studenty je třeba zkontrolovat ještě před uložením, že jich aplikace nalezla správný počet a správně jim přiřadila třídy. Studenty lze nahrávat i ručně, poté je ale nutné je přiřadit do tříd. Po vytvoření studentů a tříd je možné pokračovat s učiteli, učebnami a skupinami. Skupiny, ve kterých jsou studenti z více tříd, se přidávají na obrazovce „skupiny“, ty, v nichž jsou studenti jen z jedné třídy, se přidávají na obrazovce dané třídy. Poté už lze přejít k vytváření samotných předmětů. Ty se přidávají na obrazovkách jednotlivých tříd a skupin.

Tímto jsou všechna výchozí data uložená a na stránce učitelů lze zkontrolovat úvazky. Než se začne s tvorbou rozvrhu, je třeba zadané informace projít, protože jejich zpětnou úpravou mohou vzniknout nepodchycené kolize v rozvrhu. Vytváření rozvrhů probíhá rozklinkáváním jednotlivých polí rozvrhu a přidáváním hodin. Hodiny lze přidávat v libovolném pořadí. U každé třídy a každého učitele je možné pod tabulkou rozvrhu zkontrolovat, zda ještě zbývá přiřadit nějaké hodiny.

# 8. Seznam použitých zdrojů

## 8.1 Seznam informačních zdrojů

Dokumentace Bootstrap: Tables. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/content/tables/>

Dokumentace Bootstrap: Select. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/forms/select/>

Dokumentace Bootstrap: Grid System. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/layout/grid/>

Dokumentace Bootstrap: Modals. Upraveno 2023, cit. 20. 3. 2025. Dostupné online: <https://getbootstrap.com/docs/5.3/components/modal>

Dokumentace EF Core: Creating Model. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://learn.microsoft.com/ef/core/modeling/>

Dokumentace EF Core: Relationships. Upraveno 2023, cit. 15. 3. 2025. Dostupné online: <https://learn.microsoft.com/ef/core/modeling/relationships>

Dokumentace ASP.NET: Authentication. Upraveno 2024, cit. 20. 3. 2025. Dostupné online: <https://learn.microsoft.com/aspnet/core/blazor/security>

Wikipedia: Relational database. Upraveno 2025, cit. 25. 3. 2025. Dostupné online: <https://en.wikipedia.org/wiki/Relational_database>

VŠTE České Budějovice: Základní pojmy databáze. Upraveno 2017, cit. 27. 3. 2025. Dostupné online: <https://is.vstecb.cz/el/vste/projektclill/L_DAT/um/Databaze_e-learning.pdf>

## 8.2 Seznam zdrojů kódu

1. V rámci všech dialogových oken (Modals) byl využit upravený kód ze stránky Stack Overflow.  
   Stack Overflow: *“ How to use Bootstrap modal in Blazor client app?”* [online]. 2020, cit. 15.3.2025. Dostupné online: <https://stackoverflow.com/questions/59256798>
2. A screen shot of a computer program

   AI-generated content may be incorrect.V rámci animací pro úpravu buňky v tabulce rozvrhu (wwwroot/app.css) byl využit upravený CSS kód vygenerovaný za pomoci AI chatbotu ChatGPT ve verzi z 8. března 2025 s následující instrukcí: *“Write CSS for appearing white plus and green background in the middle of a table cell on hover, with faster disappearing”*. Vygenerovaný kód:
3. A computer code on a black background

   AI-generated content may be incorrect.Ve funkci models/Teacher.GetInitials() byl využit upravený kód vygenerovaný za pomoci AI chatbotu ChatGPT ve verzi z 15. března 2025 s následující instrukcí: *“ Write C# method, that turn string name into initial”*. Vygenerovaný kód:
4. Základní kostra souboru Program.cs je vygenerována v rámci template projektu Blazor app, obsaženého v programu Visual Studio 2022. Části pro nastavení ověřování uživatele jsou převzaty z dokumentace.

Dokumentace ASP.NET: Authentication. Upraveno 2024, cit. 20. 3. 2025. Dostupné online: <https://learn.microsoft.com/aspnet/core/blazor/security>